



TUNKU ABDUL RAHMAN UNIVERSITY COLLEGE
Faculty of Computing and Information Technology
Department of Mathematical and Data Science (DMDS)

BMCS 2113: Machine Learning

Assignment

**Title : Comparative Study of Prediction on White Wine Quality by Classification
Techniques.**

(Semester 1, Year 2021/2022)

PROG./ YEAR/ SEM/ TUTORIAL GROUP : RMM3 Y3S1 Group 2

LECTURER'S NAME : Ms. ASHIKIN BINTI ALI

STUDENT'S NAME	ID NO
Lim Wei Yang	20WMR09185
Tain Aik Siang	20WMR09189
Ong YiLiang	20WMR09187
Yong Yit Ming	20WMR09195

Table of Contents

Introduction	3
Background Study	3
Objectives	3
Methodologies	4
Data Preparation	4
Data Exploration and Preprocessing	5
Modified Dataset	9
Supervised Machine Learning Model	11
K Nearest Neighbors	11
Naive Bayesian	12
Decision Tree and Random Forest	13
Support Vector Machine	15
Logistics Regression	16
Multilayer Perceptron	17
Algorithm Tuning	18
GridSearchCV	18
RandomizedSearchCV	18
StandardScaler	18
Features Reduction	18
Model Performance	19
Experimental Results	20
Model Performance Comparison	20
Conclusion	22

Introduction

Background Study

Nowadays, the consumers enjoy wine more and more especially for the white wine. Wine industry is exploring new advances for both wine making and offering measures to back up this development. Evaluate wine certification by using the physicochemical and sensory tests. The discrimination of wines is certifiably not a simple interaction inferable from the intricacy and heterogeneity of its headspace. The classification of white wine quality has been applied by using data mining techniques. The study of the white wine quality is very important because the economic value of white wine is to protect the quality of white wine, to forbid adulteration of wine and control the processing. The aim of the machine learning is to create the models from the data to predict the white wine quality.

Objectives

1. To experiment with different classification methods to see which yields the best performance
2. To determine which features are the most indicative of a good quality wine

Methodologies

Data Preparation

The dataset is a White Wine Quality dataset that is publicly available for research purposes from Kaggle website. This dataset is also available from the [UCI machine learning repository](#), consisting 4898 instances along with 11 features and 1 output variable based on sensory data, Quality from the score 0 (very bad) to the score 10 (very excellent). The 11 features include fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol. pH describes how acidic or basic a wine is on a scale range from 2.72 - 3.82 on the pH scale. Chloride is the amount of salt in the wine while alcohol is the percentage of alcohol content of the wine.

The purpose of this dataset is to predict the quality of every wine sample, using a range of physicochemical properties. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (output) variables are available, for example there is no data about any grape types or wine brand or wine selling price. Table 1 indicates the 11 different physicochemical properties and their data statistics of the white wine dataset.

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
count	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000
mean	6.854788	0.278241	0.334192	6.391415	0.045772	35.308085	138.360657	0.994027	3.188267	0.489847	10.514267
std	0.843868	0.100795	0.121020	5.072058	0.021848	17.007137	42.498065	0.002991	0.151001	0.114126	1.230621
min	3.800000	0.080000	0.000000	0.600000	0.009000	2.000000	9.000000	0.987110	2.720000	0.220000	8.000000
25%	6.300000	0.210000	0.270000	1.700000	0.036000	23.000000	108.000000	0.991723	3.090000	0.410000	9.500000
50%	6.800000	0.260000	0.320000	5.200000	0.043000	34.000000	134.000000	0.993740	3.180000	0.470000	10.400000
75%	7.300000	0.320000	0.390000	9.900000	0.050000	46.000000	167.000000	0.996100	3.280000	0.550000	11.400000
max	14.200000	1.100000	1.660000	65.800000	0.346000	289.000000	440.000000	1.038980	3.820000	1.080000	14.200000

Table 1. The physicochemical data statistics of white wine

Data Exploration and Preprocessing

Data Exploration is a critical step to show data visualization and data understanding before fitting a model and making a prediction. Firstly, our group planned to show the histogram for all numeric type features to determine the variability of the data whether the relationship among each example is strong or not.

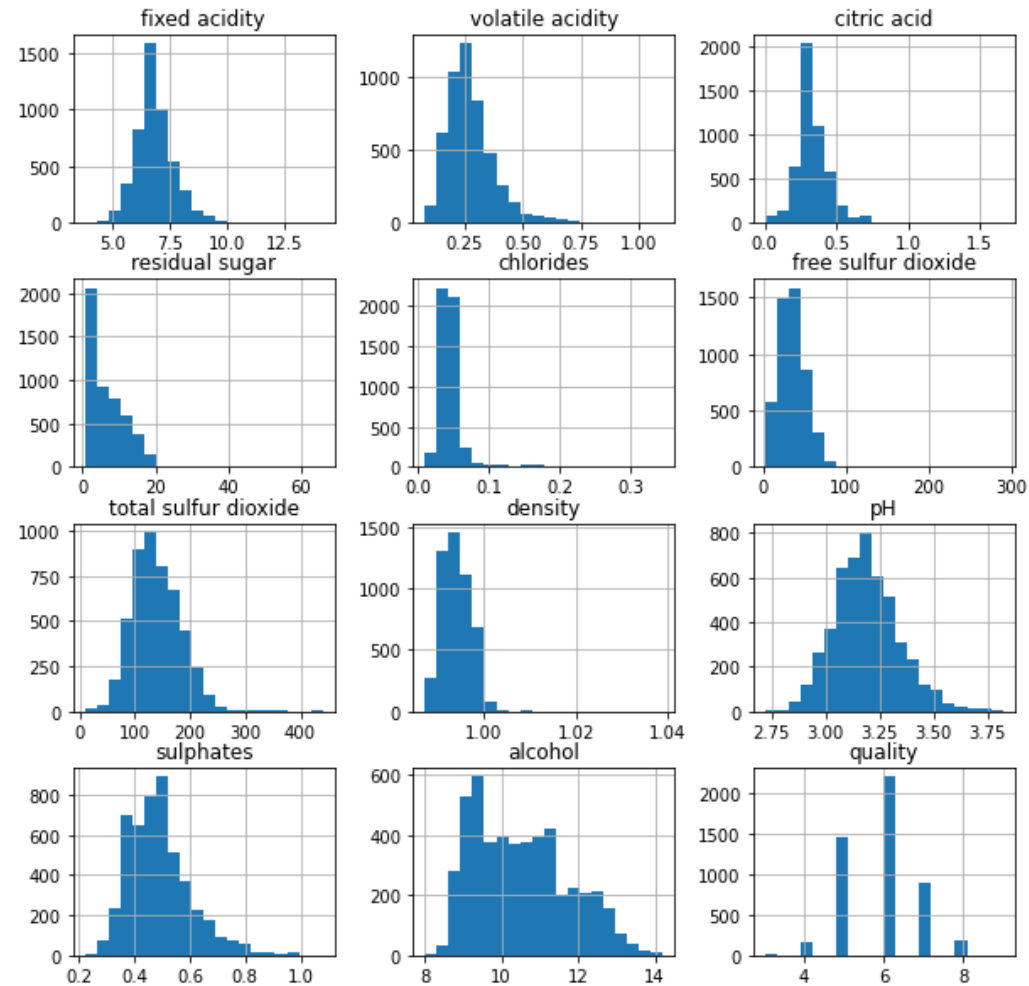


Figure 1. Histogram for each variables

Next, correlation between features and target is an important measurable value which must be calculated before modelling. After evaluating the correlation between all independent variables and dependent variables, We will reduce some features that have weak correlation on dependent variables. To evaluate the correlation between features and target, we use a bar chart to calculate the correlation for each feature to the target variable which is quality.

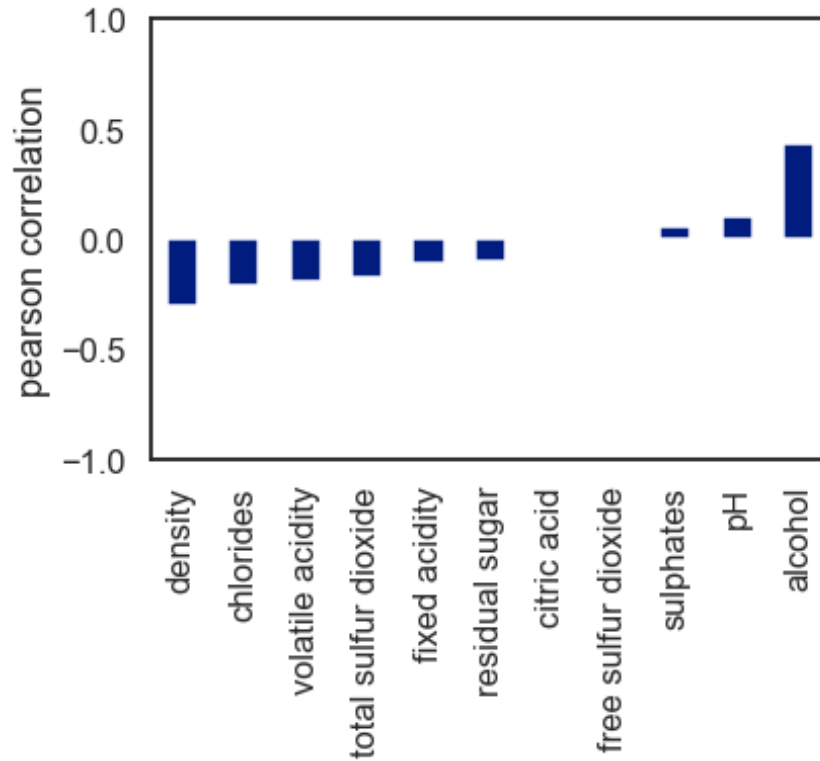


Figure 2. Pearson correlation between features and target variables

In this figure, all the attributes form a heatmap showing correlation value from -1 to 1. The darker the colour, the larger the correlation magnitude. The diagonal line of value 1 through the matrix indicates a perfect correlation of each attribute with itself. From this heatmap, we observed that density has a much stronger relationship with residual sugar and alcohol while alcohol indicates the highest relationship to Quality variable among the other attributes. And that is why the quality of a wine is usually determined by the percentage of alcohol content in the wine. Alcohol content affects a wine's body, since alcohol is more viscous than water. A wine with higher alcohol content will have a fuller, richer body, while a lower alcohol wine will taste lighter and more delicate on the palate.

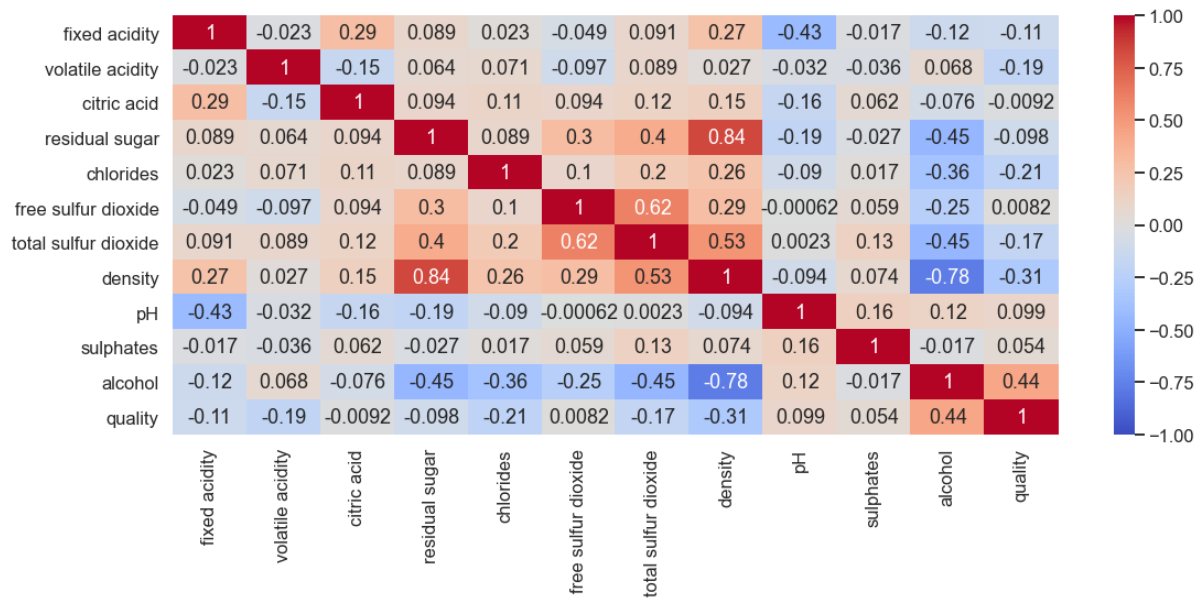


Figure 3. Correlation matrix heatmap on each features

In the figure below, a pairplot for all attributes is shown. Through the pairplot, we can clearly see the distribution of every single variable in the diagonal and the relationship between two variables on the upper or lower triangle which actually visualised the same information. Besides that, the pairplot would tell us the best set of features to form the most separated cluster too. By looking at the pattern of scattered points, we will be able to roughly investigate the unsupervised model for a better clustering and prediction that suits the features in this dataset.

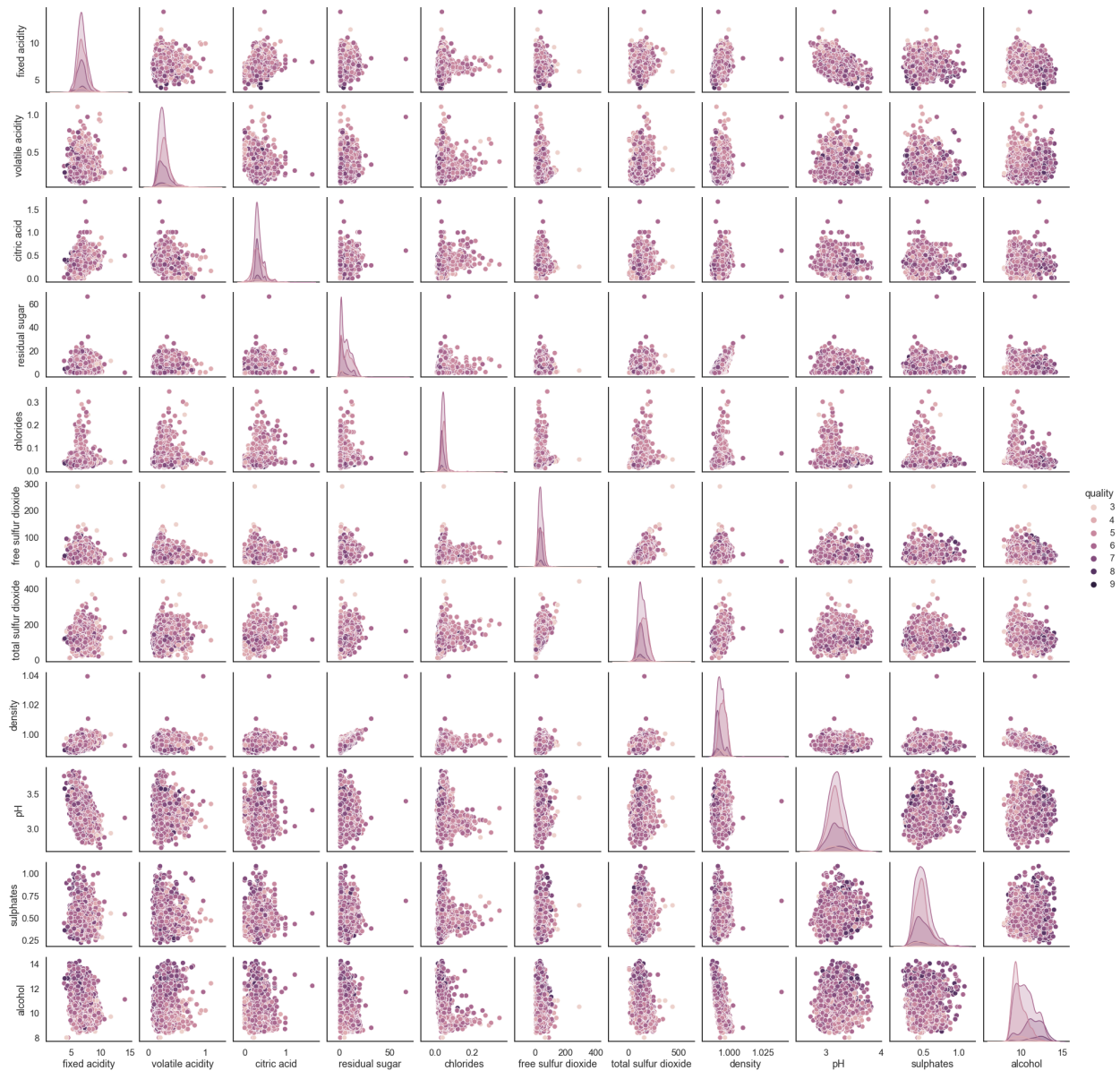


Figure 4. Pairplot of dual features class by multinomial quality

Modified dataset

For this problem that we study, since we have 7 classes in target values in the original dataset. But, our study is to classify the wine quality whether it is poor quality and good quality. From the research of wine quality, the wine quality with more or equal to 7 is in good quality, otherwise the wine is in poor quality. Therefore we modify the target column into binary class which is only good and poor. Not only that, the dataset is determined whether the dataset has missing values or illegal values. Luckily, we determined there are no missing values and illegal values have been entered in this dataset. After the data preprocessing, the correlation and the pairplot have been changed as the diagrams shown below.

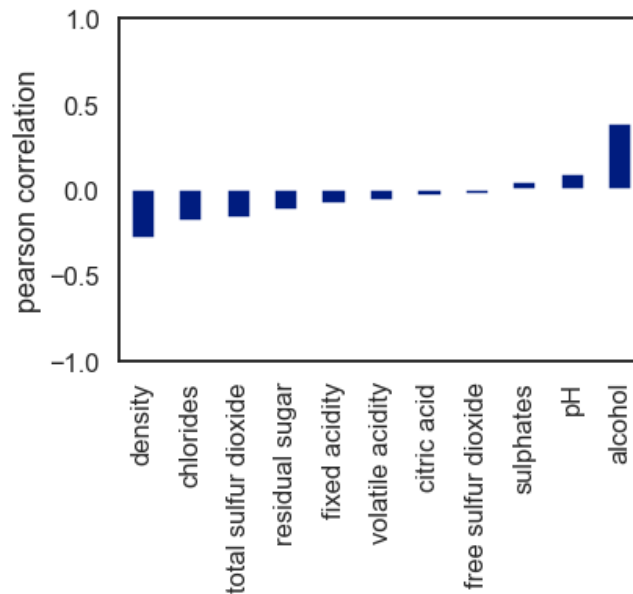


Figure 5. Pearson correlation between features and target variables after modified

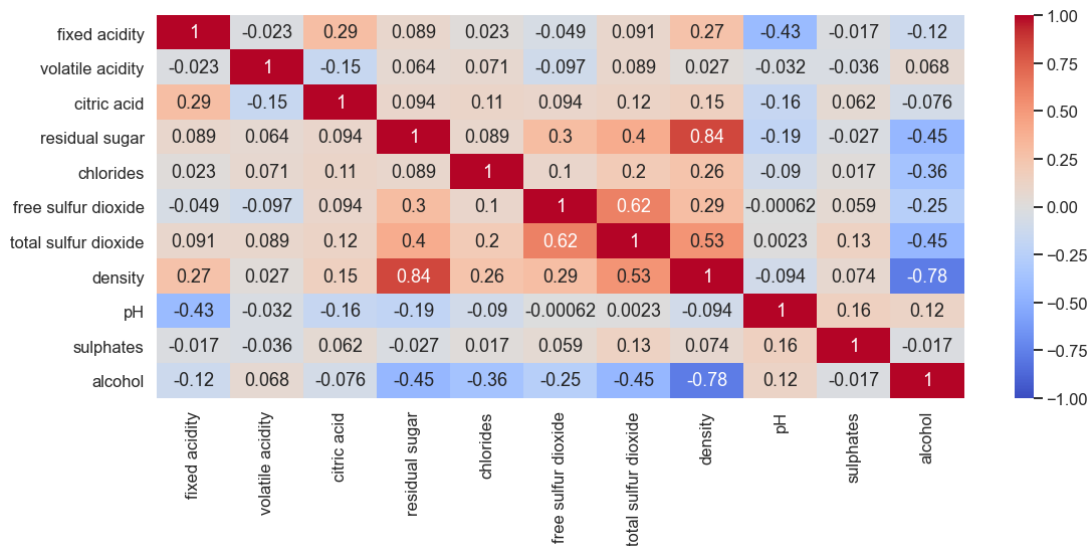


Figure 6. Correlation matrix heatmap after modified

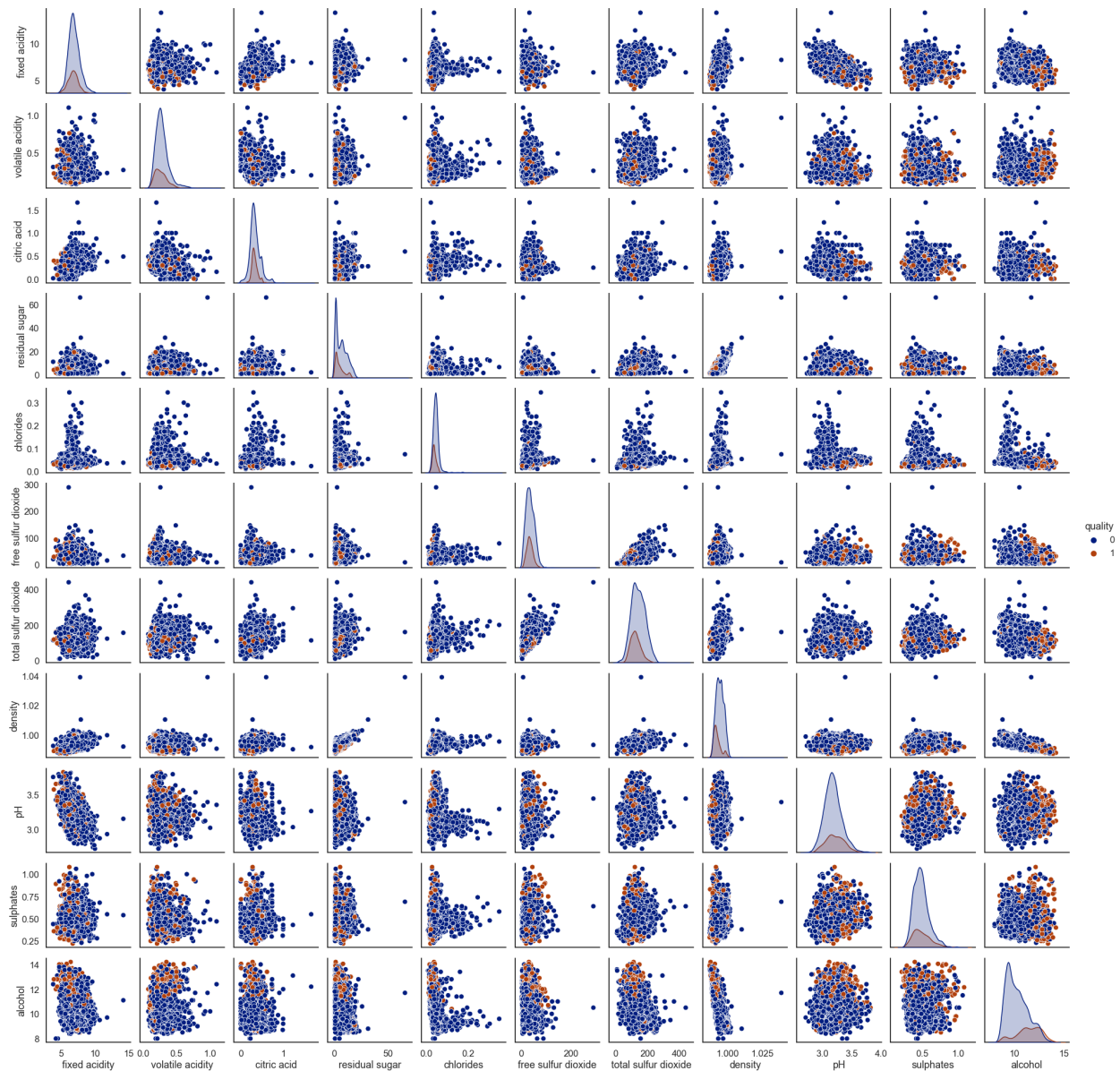


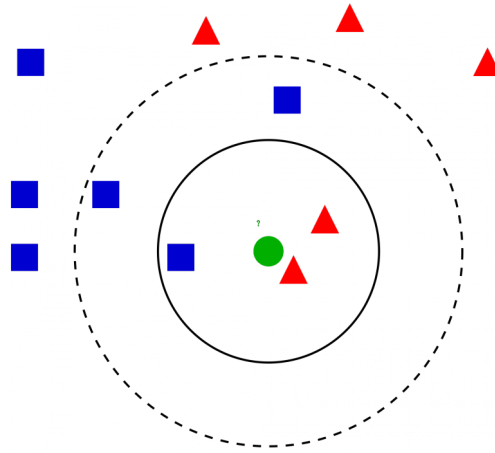
Figure 7. Pairplot between dual features class by binary quality after modified

Supervised Machine Learning Model

For this model, machine learning wants a training dataset and testing dataset to evaluate the performance of this model. For separating the dataset into train and test, we use split data function to split data into 7:3 which is the ratio of train to test in the random seed of 42. For each model, we use train data to create a model and test the performance from test data.

K Nearest Neighbors

K-nearest neighbors algorithm (k-NN) is a non-parametric supervised classification method. K-NN can be called as the most popular and laziest model to train with a dataset. Because K-NN training is to minimize the distance area around the number of K points we set. KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry.



The number of Neighbors is the main factor of the K-NN model. It is because the model training is to find the number of neighbors which is closest to selected points. For example, The model will find only one point which is closest to the selected point if $K = 1$ but It will lead model overfitting due to the error rate at $K=1$ is always zero for the training sample so the validation error will be higher in the testing sample. If K is increased, the number of neighbors to find with the closest point will increase. However, if K is very high, the model will underfitting due to the poor training rate.

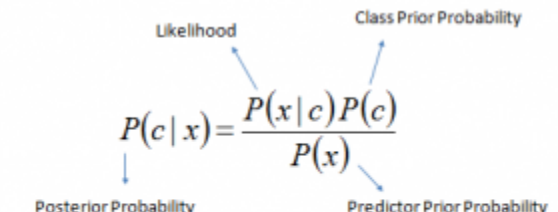
For K-NN, Distance Metrics also be the one to calculate the distance between one point to another closest point. Depending on the context of the problem, there are 2 popular distance metrics that can be used which are Euclidean Distance (Right) and Manhattan Distance (Left).

$$d(x, y) = \sum_{i=1}^n |x_i - y_i| \qquad d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Naive Bayesian

Naive Bayes algorithm is a supervised classification method and is a classification technique based on the Bayes' Theorem with the independence assumption between the features. In simple terms, Naive Bayes classifier expects that the presence of a specific feature in a class is unrelated to the presence of other features. Naive Bayes model is useful for large datasets and easy to build. It also outperformed sophisticated classification methods.

Bayes Theorem:



$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

Types of Naive Bayes Classifier:

- **Multinomial Naïve Bayes Classifier**
 - It is used for discrete, eg: counts.
- **Bernoulli Naïve Bayes Classifier**
 - It is used for the features which are binary, eg: zero & one.
- **Gaussian Naïve Bayes Classifier**
 - It is used for continuous and classification.

For this model, we choose the Gaussian Naive Bayes as our classification model, because most of our features value is continuous which is only suitable for Gaussian Naive Bayes Classifiers.

Decision Tree and Random Forest

Decision Tree is an algorithm belonging to the family of supervised learning. This algorithm can be used for solving classification and regression problems. The goal of using a Decision Tree is to create a training model that can be used to predict the class or value of the target variable by learning simple decision rules inferred from prior data. In Decision Trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.

The common problem with Decision trees is they fit a lot. If there is no limit set on a decision tree, it will give you 100% accuracy on the training data set because in the worst case it will end up making 1 leaf for each observation. Hence, one of the ways to remove overfitting is using the Random Forest.

A random forest algorithm consists of many decision trees. The 'forest' generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction as the trees protect each other from their individual errors.

Examples of tuning parameter of a treeCriterion

Entropy measures the level of impurity or the randomness in a group of examples

$$E(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

Entropy for single attribute

$$E(T, X) = \sum_{c \in X} P(c) E(c)$$

Multiple attributes

Gini is used to evaluate splits in the dataset and it performs only for binary splits.

$$Gini = 1 - \sum_{i=1}^c (p_i)^2$$

Max_depth

In general, the deeper the tree is allowed to grow, the more complex the model will become. This is due to there being more splits and it captures more information about the data. Be careful that this would be one of the root causes of overfitting in decision trees because the model will fit perfectly for the training data but will not be able to generalize well on the test set.

max_features

The number of features to consider when looking for the best split.

min_samples_leaf

Similar to min_samples_split, min_samples_leaf is also used to control over-fitting by defining that each leaf has more than one element. Thus ensuring that the tree cannot overfit the training dataset by creating a bunch of small branches exclusively for one sample each.

Support Vector Machine

Support Vector Machines (SVM) is a method that was derived from statistical learning theory by Vapnik and Chervonenkis. It was first introduced in 1992 by Boser, Guyon, and Vapnik. This method is used for the classification of both linear and nonlinear data. It uses a nonlinear mapping to transform the original training data into a higher dimension. The ideology behind it is to search for the linear optimal separating hyperplane in this new dimension. A hyperplane can separate data from two classes, with a suitable nonlinear mapping to sufficiently high dimension and the SVM uses support vectors and margins to find this hyperplane.

The equation of the hyperplane in “m” dimension can be given as

$$\begin{aligned}
 y &= w_0 + w_1x_1 + w_2x_2 + w_3x_3 \dots \\
 &= w_0 + \sum_{i=1}^m w_ix_i \\
 &= w_0 + w^T X \\
 &= b + w^T X
 \end{aligned}$$

where

w_i = vectors ($w_0, w_1, w_2, w_3, \dots, w_m$)

b = biased term (w_0)

X = variables

Logistics Regression

Logistic regression is a class of regression to predict the dependent variable by using independent variables. When the dependent variable has only two events, then it is a binary logistic regression. When the dependent variable has more than two events, then it is a multinomial logistic regression. When the dependent variable has more than two events but all have ordered in levels, then it is an ordinary logistic regression. Logistic Curve in Logistics Regression can predict the target values between 0 and 1 will be generated. So, the dataset must be in binary dataset.

For this Logistic model, we use the basic linear function to transform into a logistic function such that.

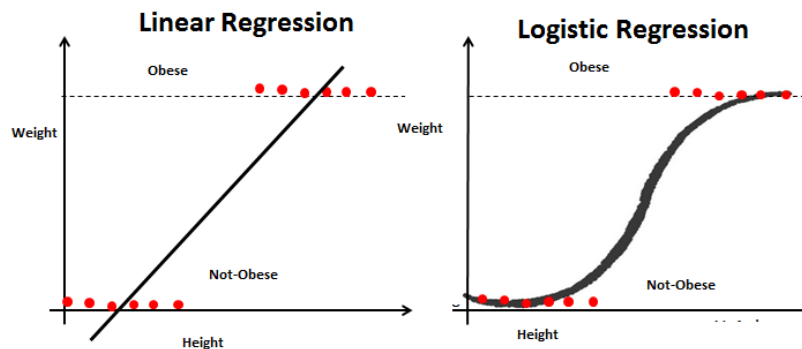
$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$$

For transformation, the sigmoid function is a S-shaped function which has the range from 0 to 1 with domain from infinity to infinity. The formula has shown below :

$$S(x) = \frac{1}{1+e^{-x}}$$

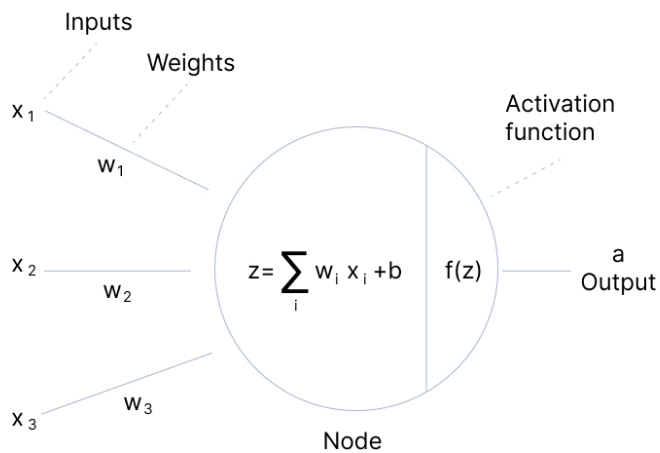
From the Sigmoid Function then we can sub Y in Linear Function into x in Sigmoid function so the logistic regression has formed.

$$p = \frac{1}{1 + e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n}}$$



Multilayer Perceptron

A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). The Neural Network can be imagined like a neuron in our brain which has an input layer, hidden layer and output layer. Input layer is the first layer which collects initial data for the neural network. Hidden layers are the middle layers which are the intermediate layer between input and output layer and place where all the computation is done. Output layer is the last layer which produces the result for given inputs. Each node is connected with each node from the next layer and each connection has a particular weight. Sometimes, each node is also connected with a bias node. Each node has an activation function which can calculate the final result in the node. Activation functions can have several types of popular activation functions which are logistics function, the rectified linear unit function and hyperbolic tangent function.



V7 Labs

Algorithm Tuning

Algorithm Tuning is a step of improvement model that can find the best points for our problems. In this study, we perform several algorithm tuning methods, namely GridSearchCV(), RandomSearchCV(), StandardScaler() and Features Reduction, to further evaluate the capability of the model and validate the model's performance after training the classifier.

GridSearchCV

GridSearchCV is the hyperparameter seeker that helps the user to tune the model parameters for reducing the rate of model overfitting. This function is created from Scikit-learn's(or SK-learn) model_selection package. This method helps to loop through predefined hyperparameters and fit the model on the training set. GridSearchCV forms all the combinations of the parameters listed in the set and evaluates the model for each combination using the Cross-Validation method. Finally, the best model parameters will be selected from the listed hyperparameters.

RandomizedSearchCV

While there is a better choice to use Grid Search when the number of parameters to be considered is particularly high and the magnitude of influence is imbalanced. RandomizedSearchCV optimized the parameters of the estimator by using a cross validation method too. This method does not try out all parameter values, but rather a fixed number of parameter settings is sampled from the specified distributions.

StandardScaler

The features that are measured at different scales do not contribute equally to the model fitting and model learned function and might end up creating bias. So, to deal with the problem features-wise standardized with converting into standard normal distribution is usually used to model fitting.

Features Reduction

Features reduction removes multicollinearity resulting in improvement of the model in use. By removing the features that do not have a strong relationship with the target. According to figure 5, we use pearson correlation to figure out which features have a strong relationship with the binary target. For example, we drop the features and leave only features with the higher correlation which perform the best accuracy in our model and reduce the training process duration.

At the end by using these several methods, we choose the model which has the highest accuracy as the optimal model trained. After selecting the optimal model for each supervised learning, we will make a comparison between each model based on the model performance.

Model Performance

Confusion Matrix is a performance measurement for the machine learning classification problem where output can be two or more classes. It is a table with four different combinations of predicted and actual values.

The 4 combinations of predicted and actual values are:

- True Positive (TP) which is positive predicted and true results.
- False Positive (FP) which is positive predicted but false results. (Type 1 Error)
- True Negative (TN) which is negative predicted and true results.
- False Negative (FN) which is negative predicted but false results. (Type 2 Error)

Confusion Matrix		Predicted Values	
		Negative	Positive
Actual Values	Negative	TN	FP
	Positive	FN	TP

The uses of the Confusion Matrix

Confusion Matrix is useful for measuring Recall, Precision, Accuracy and F-Measure.

1. Recall

Recall is the ratio of correctly predicted positive observations to all observations in actual class.

$$\text{Recall} = \frac{TP}{TP + FN}$$

2. Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. High precision relates to the low false-positive rate.

$$\text{Precision} = \frac{TP}{TP + FP}$$

3. Accuracy

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

4. F-Measure

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar costs. It's better to look at both Precision and Recall if the difference between false positive and false negative is too big.

$$\text{F - Measure} = \frac{2 * \text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}}$$

Experimental Results

Model Performance Comparison

First, the original model which default parameter also be used to make the first run on model performance evaluation. The table below shows the model performance without algorithm tuning.

For Good Quality

Classifier	Accuracy	Precision	Recall	F1-Score
K-Nearest Neighbor	77.55%	49.79%	35.26%	41.28%
Naives Bayes	70.68%	40.79%	68.69%	51.19%
Decision Tree	82.79%	61.24%	62.92%	62.07%
Random Forest	88.37%	81.35%	62.31%	70.57%
Support Vector Machine	77.62%	0%	0%	0%
Logistics Regression	79.39%	61.82%	20.67%	30.98%
Multilayer Perceptron	78.50%	59.15%	12.77%	21.00%

For Poor Quality

Classifier	Accuracy	Precision	Recall	F1-Score
K-Nearest Neighbor	77.55%	82.78%	89.75%	86.12%
Naives Bayes	70.68%	88.76%	71.25%	79.05%
Decision Tree	82.79%	89.22%	88.52%	88.87
Random Forest	88.37%	89.82%	95.88%	92.75%
Support Vector Machine	77.62%	77.62%	100%	87.40%
Logistics Regression	79.39%	80.81%	96.32%	87.88%
Multilayer Perceptron	78.50%	79.49%	97.46%	87.56%

Next, we use some algorithm tuning listed on methodology to reduce the model overfitting. The table below shows the model performance with algorithm tuning.

For Good Quality

Classifier	Accuracy	Precision	Recall	F1-Score
K-Nearest Neighbor	86.94%	77.96%	58.05%	66.55%
Naives Bayes	70.61%	40.65%	68.09%	50.91%
Decision Tree	81.02%	58.93%	50.15%	54.19%
Random Forest	88.16%	82.70%	59.57%	69.26%
Support Vector Machine	86.87%	78.81%	56.53%	65.84%
Logistics Regression	79.12%	59.65%	20.67%	30.70%
Multilayer Perceptron	80.61%	63.10%	32.22%	42.66%

For Poor Quality

Classifier	Accuracy	Precision	Recall	F1-Score
K-Nearest Neighbor	86.94%	88.73%	95.27%	66.55%
Naives Bayes	70.61%	88.57%	71.34%	79.03%
Decision Tree	81.02%	86.22%	89.92%	88.03%
Random Forest	88.16%	89.21%	96.41%	92.67%
Support Vector Machine	86.87%	88.41%	95.62%	91.87%
Logistics Regression	79.12%	80.75%	95.97%	87.71%
Multilayer Perceptron	80.61%	82.87%	94.57%	88.33%

Conclusion

For each classification model, we analyzed how the results vary after the use of algorithm tuning methods. The study includes the descriptive statistics of the dataset, the analysis of classifiers and lastly the results of each model's before and after performance. The results are described in percentage of correctly classified instances, accuracy, precision, recall and F1-score before and after algorithm tuning. Classifiers such as K-Nearest Neighbors, Naive Bayes, Decision Tree and Random Forests, Support Vector Machines, Logistics Regression and Multilayer Perceptron are used to evaluate the white wine quality dataset. Results from the experiment indicate that Random Forest yields the best performance among the classifiers in this classification task, achieving a high accuracy score of 88%, whereas Naive Bayes performs the worst with only an accuracy score of 70%. We also successfully determined that features including alcohol, density, chlorides, total sulphur dioxide and residual sugar have much higher correlation with the target variable, where alcohol and density will be the most indicative features of a good quality wine. However, more work can be done to further increase the quality of prediction by exploring other machine learning models. In our future work, we will validate our results with other related real-life cohorts.